

## Assignment 7 – Your First Model

---

**Due Date:** Tuesday March 4, 2008 at 11:55PM

In this assignment, you will create your first "Model". The Model is the part of the MVC architecture that deals with persistent data such as data that is stored in a database.

This assignment builds on the previous assignment - you may want to make a copy of the previous assignment to a new folder. As long as you copy the entire contents of the folder the Rails application will work just fine in the new folder.

As you go through this assignment - you can make you own model - you do not have to follow my example precisely - my model will be about members. The book covers a different but very similar model called "Story".

In this assignment we will first set up our model and then interact with the data model in the interactive Rails console. Then we will make changes to our controller to take data from our form and insert it into the model.

In Rails, we use an approach called ActiveRecord. ActiveRecord allows us to work with objects that represent the data to be stored in the database tables. All we have to do is tell ActiveRecord the kinds of information we would like to store and it handles the database creating, reading, and writing without us every having to see the SQL (Structured Query Language) that is generated.

It is very important that you read pages 97-99 in the textbook. Stop reading on page 99 at the paragraph that starts with "So, lets create.." If you read page 100 - you will be confused! You can look at it - but it has \*nothing\* at all to do with this assignment.

Then read pages 118-129. However as you read, ignore everything that is said about stories.yml and YAML - this is good stuff but not necessary for this assignment.

Your assignment is to create and populate one table in your database and look into your database to make sure the data is there, and then hand in your completed database file. Then we will add code to your controller class to add new users from the web interface.

### Part I - The Members Table

Once you have your application, lets start it up just to make sure it is a fully working copy of Assignment 6.

```
cd assn7
ruby script/server
```

Navigate to your controller <http://localhost:3000/One/> - substitute the name of your controller for "One" in the url. Click through and verify that your application is functioning - taking a short time

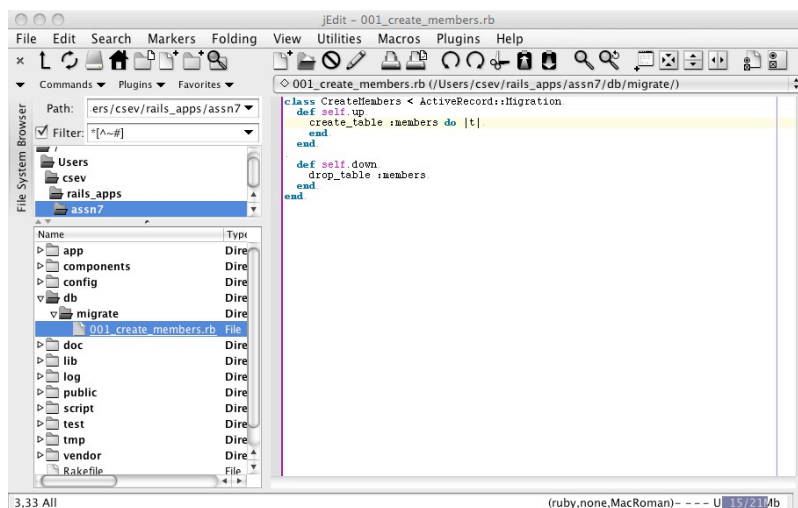
to test your application before we go off and modify it will save you frustration later when things start to break.

Once you are satisfied that your application is working - shut down your server by pressing CTRL-C.

Next, make sure you are in the assn7 directory and type the command:

```
$ ruby script/generate model member
exists app/models/
exists test/unit/
exists test/fixtures/
create app/models/member.rb
create test/unit/member_test.rb
create test/fixtures/members.yml
create db/migrate
create db/migrate/001_create_members.rb
```

This creates a model in your application by creating some files in the application. We will edit these files. Open the file `db\001_create_members.rb`. This is the "migration" file - it builds the database and gives the database structure. This is what the file looks like originally as created by the `script/generate` command. This database has no columns.



Modify the text (page 124) and add two columns to your database table:

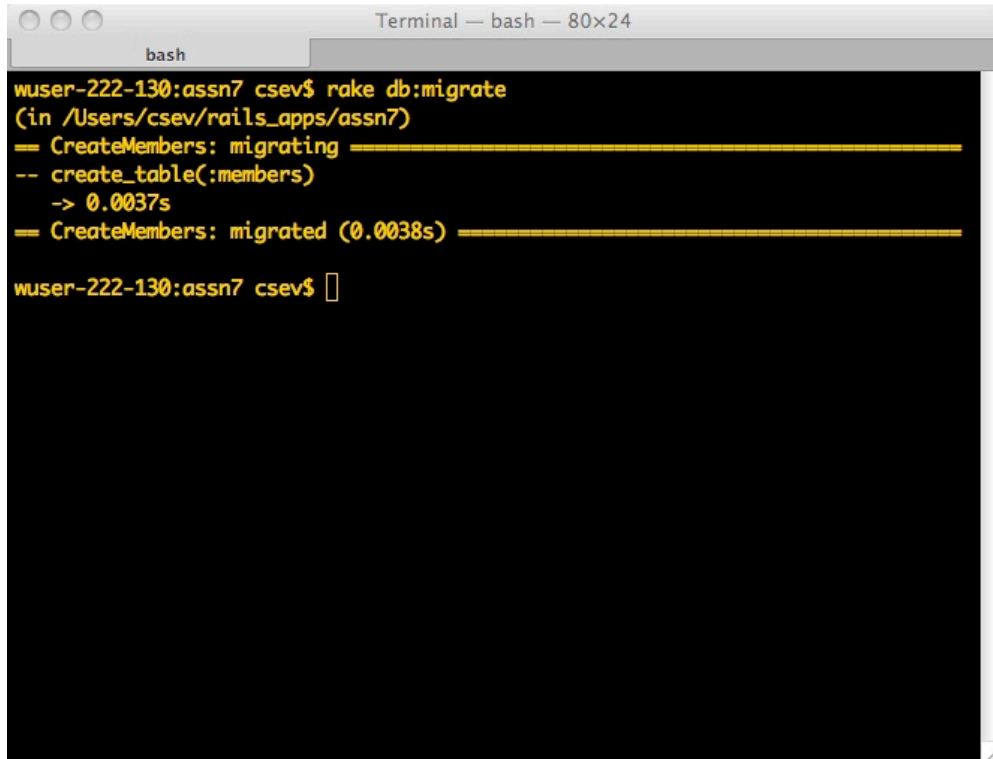
```
class CreateMembers < ActiveRecord::Migration
  def self.up
    create_table :members do |t|
      t.column :name, :string
      t.column :email, :string
    end
  end

  def self.down
    drop_table :members
  end
end
```

Next we need to run a command in the terminal window again. Make sure that you are in your application directory and then run the command:

```
rake db:migrate
```

It should look as follows:

A screenshot of a terminal window titled "Terminal — bash — 80x24". The terminal shows the command "rake db:migrate" being executed in the directory "/Users/csev/rails\_apps/assn7". The output indicates that the "CreateMembers" migration is being applied, specifically the "create\_table(:members)" step, which takes 0.0037 seconds. The final output is "CreateMembers: migrated (0.0038s)". The prompt then returns to "wuser-222-130:assn7 csev\$".

```
wuser-222-130:assn7 csev$ rake db:migrate
(in /Users/csev/rails_apps/assn7)
== CreateMembers: migrating =====
-- create_table(:members)
   -> 0.0037s
== CreateMembers: migrated (0.0038s) =====

wuser-222-130:assn7 csev$
```

If you get errors, work on the errors and fix them until the rake db:migrate command works successfully. There are two types of errors that you might encounter: (1) syntax errors in the migration file such as forgetting a comma or colon or (2) if it complains about mysql you forgot to do the -d sqlite3 on the rails command. Most of you will **not** encounter a mysql error because you have been typing -d sqlite3 on all of your Rails commands - but if you do, edit the file

```
config/database.yml
```

Replacing its contents with this:

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  timeout: 5000
test:
  adapter: sqlite3
  database: db/test.sqlite3
  timeout: 5000
production:
```

```
adapter: sqlite3
database: db/production.sqlite3
timeout: 5000
```

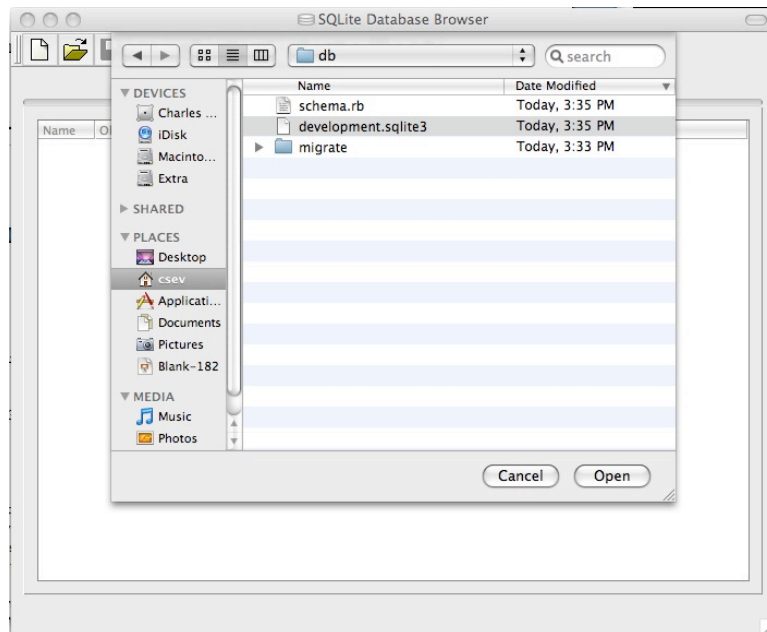
Note: If the rake command works but later you want to start over with a fresh database - perhaps after making a change to the migration script, use the command

```
rake db:migrate VERSION=0
```

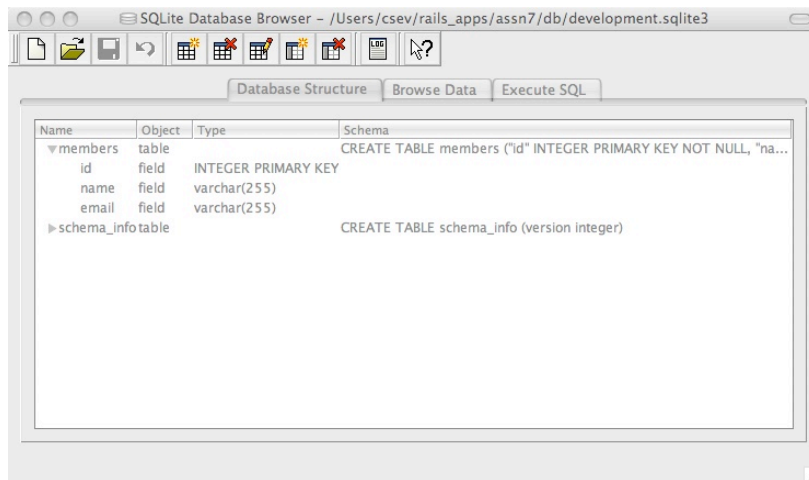
as shown on page 127 and then do another rake db:migrate to re-create an empty database (page 127 is wrong - there should not be a dash)

Next you will use the Sqlite3 browser. It is in a subdirectory in your rte-mac or rte-win directory that came with your installation. Open the Browser and navigate to the file

`\db\development.sqlite3`



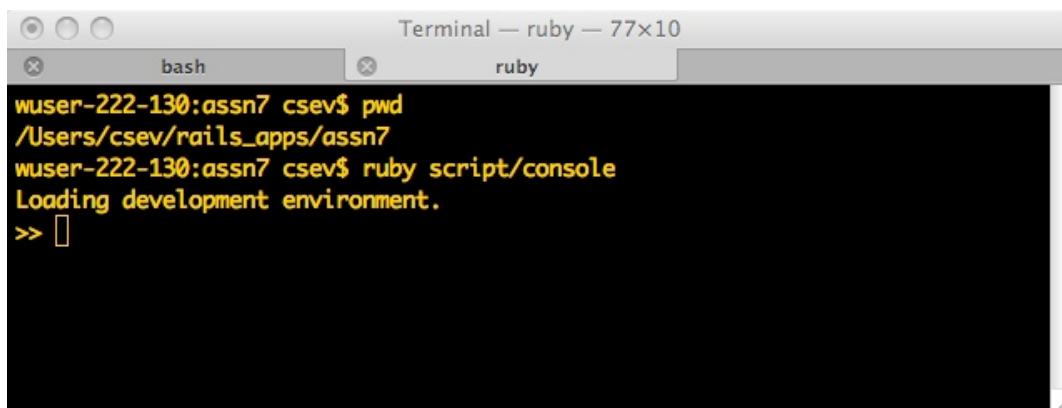
If you look at the members table, you will see that it has three columns: id, name, and email. If you see this table with these columns - it means that your **rake db:migrate** worked. Take a screen shot of your SQLite Browser showing the new table and its columns to turn in for the assignment.



The next steps we will be done in the Rails console. The Rails console is different than the Ruby interpreter (irb). The Ruby interpreter only understands the Ruby language - it knows nothing about Rails. The Rails console is started with the command `ruby script/console` (page 128). In essence the Rails console allows you to type Controller code in interactively.

Make sure you are in the right directory and your server is down and then type

`ruby script/console`



We are going to perform all of the steps on page 128 - except use our members (or whatever you are making) table instead of the stories table. Here is a log of that set of steps:

```
wuser-222-130:assn7 csev$ pwd
/Users/csev/rails_apps/assn7
wuser-222-130:assn7 csev$ ruby script/console
Loading development environment.
>> s = Member.new
=> #<Member:0x22b6b08 @new_record=true, @attributes={"name"=>nil,
"email"=>nil}>
>> s.name = "Chuck"
=> "Chuck"
>> s.email = "csev@umich.edu"
=> "csev@umich.edu"
>> s.save
```

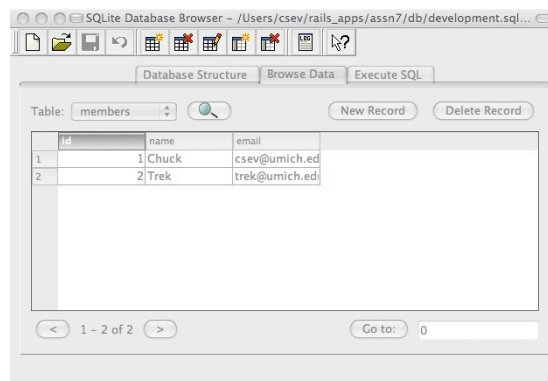
```

=> true
>> s.id
=> 1
>> s.new_record?
=> false
>> Member.count
=> 1
>> Member.create(:name => "Trek", :email => "trek@umich.edu")
=> #<Member:0x221fde8 @new_record=false, @attributes={"name"=>"Trek",
"email"=>"trek@umich.edu"},
@errors=#<ActiveRecord::Errors:0x221f460 @errors={} ,
@base=#<Member:0x221fde8 ...>>
>> Member.find(2)
=> #<Member:0x221bd88 @attributes={"name"=>"Trek", "id"=>"2",
"email"=>"trek@umich.edu"}>
>> Member.find(:all)
=> [#<Member:0x22193a8 @attributes={"name"=>"Chuck", "id"=>"1",
"email"=>"csev@umich.edu"}>, #<Member:0x2219380
@attributes={"name"=>"Trek", "id"=>"2", "email"=>"trek@umich.edu"}>]
>> Member.find(:all).last
=> #<Member:0x2216414 @attributes={"name"=>"Trek", "id"=>"2",
"email"=>"trek@umich.edu"}>
>> Member.find(:first, :order => 'id DESC')
=> #<Member:0x22128dc @attributes={"name"=>"Trek", "id"=>"2",
"email"=>"trek@umich.edu"}>
>> Member.find_by_name("Chuck")
=> #<Member:0x220fc7c @attributes={"name"=>"Chuck", "id"=>"1",
"email"=>"csev@umich.edu"}>
>>

```

Take a screenshot of the terminal window where you typed these commands. Save this screen shot for hand-in.

Go back to the SQLite Browser and re-open the development.sqlite3 file. If you look at the members table in Browse Data it should look as follows - take a screen shot of this view of your database to be handed in.



The screenshot shows the SQLite Database Browser interface. The 'members' table is selected, and the 'Browse Data' tab is active. The table contains two records:

id	name	email
1	Chuck	csev@umich.edu
2	Trek	trek@umich.edu

At the bottom of the window, it shows '1 - 2 of 2' records and a 'Go to:' field with the value '0'.

Continue to do the commands in the book through the bottom of page 133. As often as you like, you can look at the contents of the database to verify your operations by reopening the file in SQLite Database Browser.

## Part II - Adding an entry to the Table from the Web

In the previous steps you were running Rails commands in the Rails console - now we will do the same thing in your controller.

Start up your server using the command:

```
ruby script/server
```

Make sure that your application continues to work.

Open the file

```
app\controllers\one_controller.rb
```

Replace the thanks action method with the following:

```
def thanks
  logger.info "Welcome to the thanks action in the controller"
  logger.info params[:yourname]
  logger.info params[:youremail]
  memb = Member.create()
  memb.name = params[:yourname]
  memb.email = params[:youremail]
  memb.save
  @barcelona = memb.id
end
```

Then edit the file app\views\one\thanks.rhtml and change the form submission to be a post

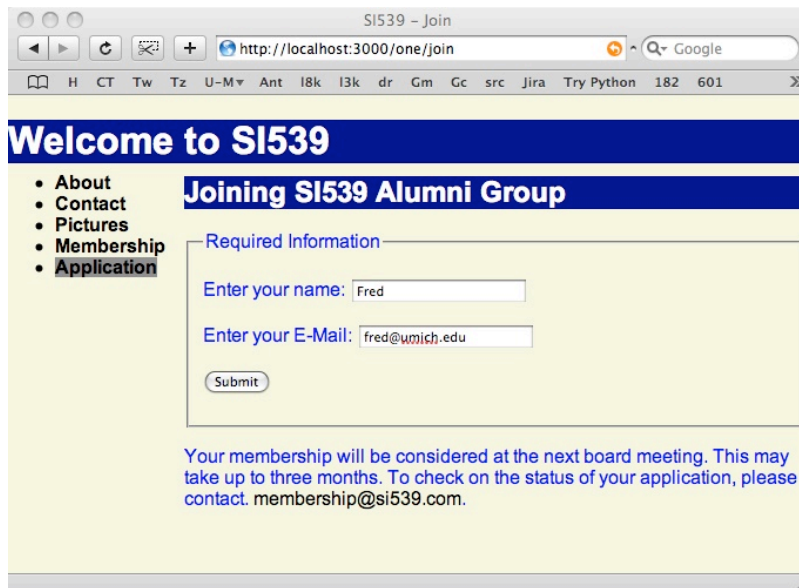
```
<form method="post" action="<%= url_for :action => "thanks" %>">
```

and the line with @barcelona to be

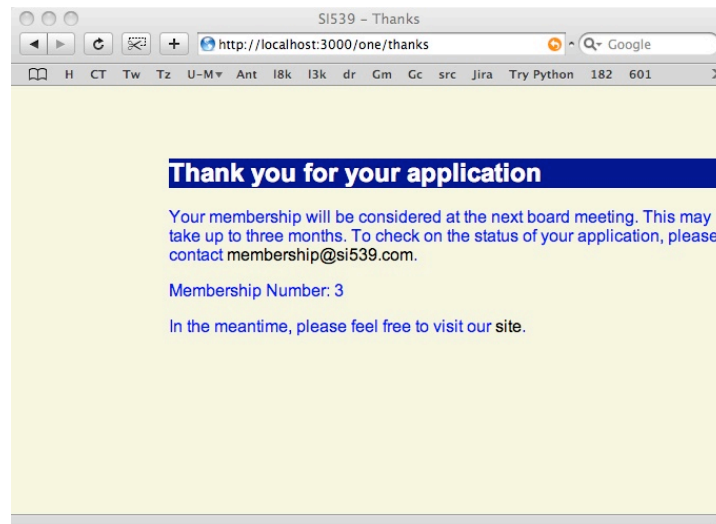
```
<p>Membership Number: <%= @barcelona %></p>
```

Since now this will be the membership number.

Then navigate to your join screen, fill in some data and press submit. Take a screen shot of the filled in data before you press submit.



Your submission should be successful and tell you the new membership number as shown below. Take a screen shot of the successful submission with the member number.



Note that because we switch to sending the parameters on the form using the POST method, they no longer show up in the URL. GET and POST work equivalently but for this, POST is more appropriate because it is modifying data.

If you look in your log (or in Wintail) you should see something like this:

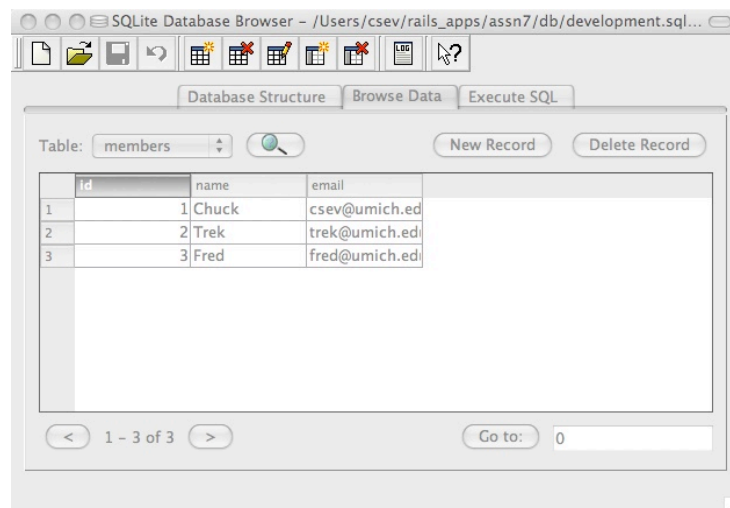


```
Terminal — ruby — 78x23
ruby

Processing OneController#thanks (for 127.0.0.1 at 2008-02-19 16:11:43) [GET]
  Session ID: b55a9ed1328c50af487ef937aa8245f6
  Parameters: {"action"=>"thanks", "controller"=>"one", "yourname"=>"Fred", "yourmail"=>"fred@umich.edu"}
Welcome to the thanks action in the controller
Fred
fred@umich.edu
SQL (0.000292) INSERT INTO members ("name", "email") VALUES(NULL, NULL)
Member Update (0.000426) UPDATE members SET "email" = 'fred@umich.edu', "name" = 'Fred' WHERE "id" = 3
Rendering one/thanks
Completed in 0.02078 (48 reqs/sec) | Rendering: 0.00253 (12%) | DB: 0.00072 (3%) | 200 OK [http://localhost/one/thanks?yourname=Fred&yourmail=fred%40umich.edu]
[]
```

Note that the WinTail sadly does not show the pretty colors - the attempt to make pretty colors just looks like gibberish in WinTail - do not worry about this. Take a screenshot of the log or Wintail showing the SQL Insert Statement as well for hand-in.

Go into the SQLite Browser and reopen the file and look at the member table.



You should see the new record. Take a screenshot of this as well for hand in.

### Part III - Wrap up and hand-in

Upload all of the screenshots to CTools mentioned above - also upload the file

\logs\development.log

This seems like a complex assignment - but it is really pretty simple - the goal is to just get you used to flipping around between your windows so you can check to see if your code is working.